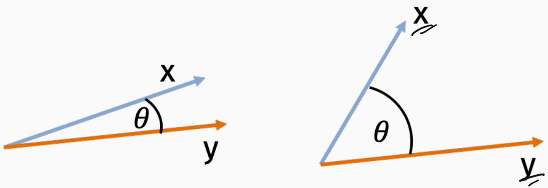
CS-GY 9223 I: Lecture 5
Gradient Decent and Its Many Forms

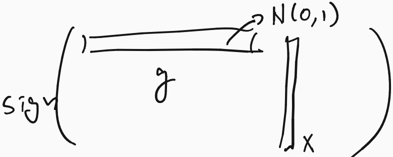
NYU Tandon School of Engineering, Prof. Christopher Musco

Cosine similarity $\cos(\theta(\mathbf{x}, \mathbf{y})) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$:



$$-1 \leq \cos(\theta(\mathbf{x}, \mathbf{y})) \leq 1.$$

SIMHASH



Locality sensitive hash for cosine similarity:

- Let $\mathbf{g} \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0, 1)$.
- $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ is definied $h(\mathbf{x}) = \text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle)$.

If $\cos(\theta(\mathbf{x}, \mathbf{y})) = v$, what is $\Pr[h(\mathbf{x}) == h(\mathbf{y})]$?

SIMHASH ANALYSIS

Dimension : 2 = d

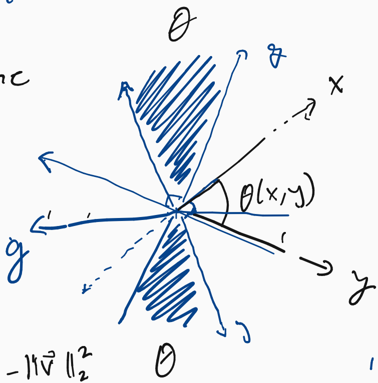
g = random gaussian

Rotation Invariance

$h(x) = \text{sign}(g^T x) = -1$

$h(y) = \text{sign}(g^T y) = 1$

$h(x) \neq h(y)$



$h(x) = -1$

$h(y) = -1$

$h(x) = h(y)$

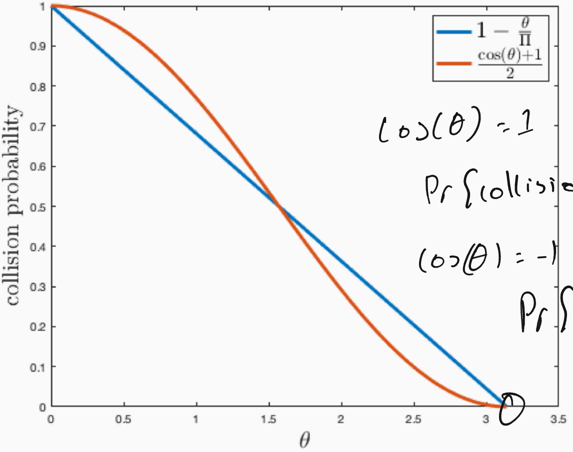
$\Pr[h(x) = h(y)]$

$\Pr(\vec{g} = \vec{v}) \sim e^{-\|\vec{v}\|_2^2}$

$\Pr(g_i = v) \sim e^{-v^2}$

$\Pr[h(x) \neq h(y)] = \frac{2\theta}{2\pi} = \frac{\theta}{\pi}$

$\Pr[h(x) = h(y)] = 1 - \theta/\pi$



$\cos(\theta) = 1$

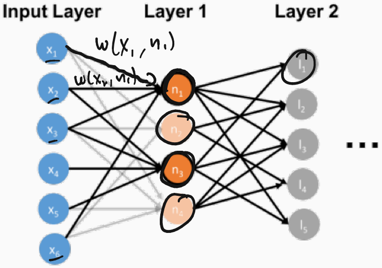
$\Pr[\text{collision}] = 1$

$\cos(\theta) = -1$

$\Pr[\text{collision} = 0]$

SIMHASH TO SPEEDUP NEURAL NETWORKS

Work of Anshumali Shrivastava at Rice University and coauthors.



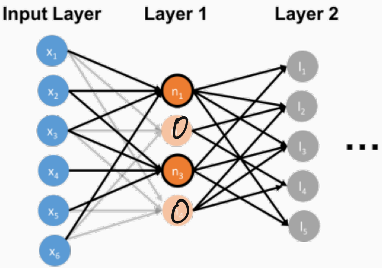
...

$$n_i = \sigma\left(\underbrace{\sum_{j=1}^{m} w(x_j, n_i)\cdot x_j}\right) = \sigma(\langle w_i, x\rangle)$$

$$\begin{bmatrix} w(x_1, n_1) \\ w(x_2, n_1) \\ \vdots \\ w(x_6, n_1) \end{bmatrix}$$

- Number of multiplications to evaluate $\mathcal{N}(\mathbf{x})$:
  $(|\mathbf{x}| \cdot |\text{layer 1}|) + |\text{layer 1}| \cdot |\text{layer 2}| + |\text{layer 2}| \cdot |\text{layer 3}| + \ldots$.
- For an approximate solution, only consider neurons on each each with high activation.

Work of Anshumali Shrivastava at Rice University and coauthors.



$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i) \cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

w_1

w_2

$\vdots$

w_n

- High activation = large value of $\sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle)$.
- Typically $\sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle)$ increases as $\langle \mathbf{w}_i, \mathbf{x}\rangle$ increases.
- Use LSH/SimHash to quickly find all $\mathbf{w}_i$ for which $\langle \mathbf{w}_i, \mathbf{x}\rangle$ is large and only include these terms in the sum.

Optimization

Have some function $f : \mathbb{R}^d \rightarrow \mathbb{R}$. Want to find $\hat{\mathbf{x}}$ such that:

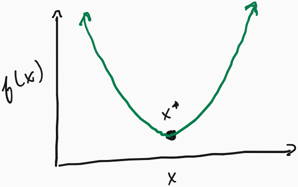
$$f(\hat{\mathbf{x}}) = \min_{\mathbf{x}} f(\mathbf{x}). \quad + \varepsilon$$

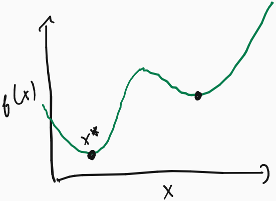
Or at least $\hat{\mathbf{x}}$ is close to a minimum. E.g. $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}} f(\mathbf{x}) + \epsilon$

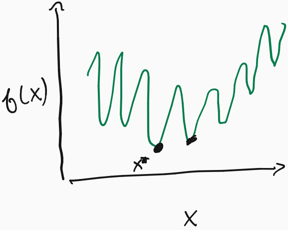
Often we have some additional constraints:

- $\mathbf{x} > 0$
- $\|\mathbf{x}\|_2 \leq R$, $\|\mathbf{x}\|_1 \leq R$
- $\mathbf{a}^T\mathbf{x} > c$.

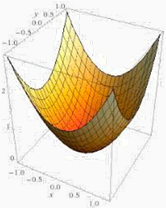
Dimension $d = 1$:

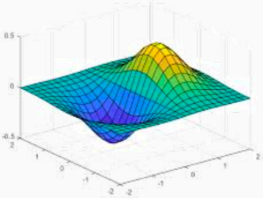


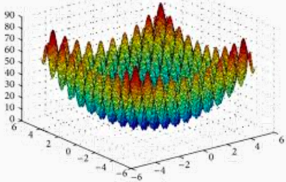




Dimension $d = 2$:







Machine learning: Want to learn a model that maps input

- numerical data vectors
- images, video
- text documents

to prediction

- numerical value (probability stock price increases)
- label (is the image a cat? does the image contain a car?)
- decision (turn car left, rotate robotic arm)

Let $M_{\mathbf{x}}$ be a model with parameters $\mathbf{x} = \{x_1, \ldots, x_d\}$.

Example:

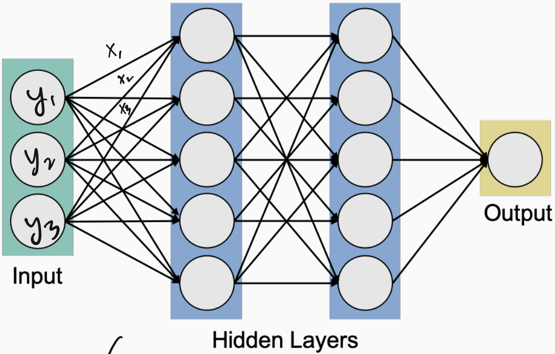
data vector

$$M_{\mathbf{x}}(\mathbf{y}) = \underbrace{\mathrm{sign}(\mathbf{y}^T\mathbf{x}_{1:d-1} + \mathbf{x}_d)}_{} \rightarrow \{-1, 1\}$$

param. vector

$X \rightarrow$ weights

Example:



M_X

Classic approach in supervised learning: Find a model that works well on data that you already have the answer for (labels, values, classes, etc.).

- Model $\boxed{M_{\mathbf{x}}}$ parameterized by a vector of numbers $\mathbf{x}$.
- Dataset $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}$ with outputs $o^{(1)}, \ldots, o^{(n)}$.

Want to find $\hat{\mathbf{x}}$ so that $M_{\hat{\mathbf{x}}}(\mathbf{y}^{(i)}) \approx o^{(i)}$ for $i \in 1, \ldots, n$.

How do we turn this into a function minization problem?

$\min f \qquad f : \mathbb{R}^d \to \mathbb{R}$

Loss function $L(M_{\mathbf{x}}(\mathbf{y}), o)$: Some measure of distance between prediction $M_{\mathbf{x}}(\mathbf{y})$ and true output o. Increases if they are further apart.

- Squared (ℓ_2) loss: $|M_{\mathbf{x}}(\mathbf{y}) - o|^2$
- Absolute deviation (ℓ_1) loss: $|M_{\mathbf{x}}(\mathbf{y}) - o|$
- Hinge loss: $1 - o \cdot M_{\mathbf{x}}(\mathbf{y})$ $\quad o, M_x(y) \in \{-1, 1\}$ $\quad$ Loss = 0 / Loss = 2
- Cross-entropy loss (log loss).
- Etc.

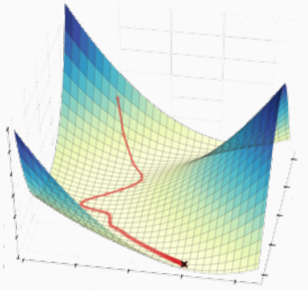
Empirical risk minimization:

$$f(\mathbf{x}) = \sum_{i=1}^{n} L\left(M_{\mathbf{x}}(\mathbf{y}^{(i)}), o^{(i)}\right)$$

Solve the optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$.

Choose $\hat{x}$ s.t. $f(\hat{x}) \leq \min_{x} f(x) + \varepsilon$

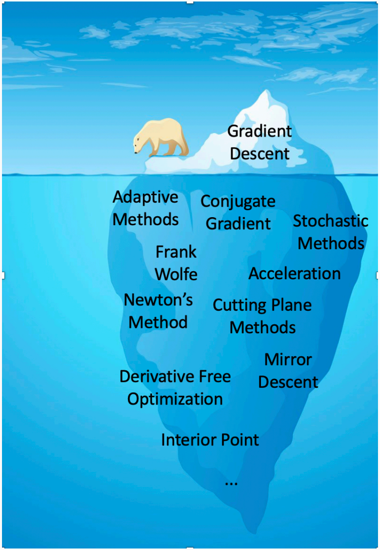
Gradient descent: A greedy algorithm for minimizing functions of multiple variables that often works amazingly well.



Abraham Maslow:

"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

So much to learn!



For $i = 1, \ldots, d$, let x_i be the i^{th} entry of $\mathbf{x}$. Let $\mathbf{e}^{(i)}$ be the i^{th} standard basis vector.

$[0\ 0\ 0\ \underset{i}{1}\ 0\ 0\ 0]$

$f(x+te^{(i)}) - f(x)$

Partial derivative:

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{e}^{(i)}) - f(\mathbf{x})}{t}$$

t close to 0

$$\frac{f(x+te^{(i)}) - f(x)}{} \approx \frac{df}{dx_i} \cdot t$$

Directional derivative:

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t}$$

$x \to x + tv \qquad x \to x + tv_1 e_1 \to x + tv_1 e_1 + tv_2 e_2$

Gradient: $f(x) \to f(x+tv)$

$\to x + tv_1 e_1 + \dots + tv_d e_d$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_d}(\mathbf{x}) \end{bmatrix}$$

$f(x) \to f(x) + tv_1 \frac{\partial f}{\partial x_1}$

$f(x) \to f(x) + tv_1 \frac{\partial f}{\partial x_1} + tv_2 \frac{\partial f}{\partial x_2}$

Directional derivative:

$f(x) \to \underbrace{f(x) + tv_1 \frac{\partial f}{\partial x_1} + \dots + tv_d \frac{\partial f}{\partial x_d}}$

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t} = \boxed{\nabla f(\mathbf{x})^T \mathbf{v}.}$$

$t \, \nabla f(x)^T v$

$f(x+tv) - f(x) \approx t \cdot D_v f(x)$

Given a function f to minimize, assume we can:

- **Function oracle**: Evaluate $f(\mathbf{x})$ for any $\mathbf{x}$.
- **Gradient oracle**: Evaluate $\nabla f(\mathbf{x})$ for any $\mathbf{x}$.

Linear least-squares regression:

- Given $\underline{\mathbf{y}^{(1)}, \ldots \mathbf{y}^{(n)}} \in \mathbb{R}^d$, $\underline{b^{(1)}, \ldots b^{(n)}} \in \mathbb{R}$.
- Want to minimize $f(\mathbf{x}) = \sum_{i=1}^{n} \left(\underline{\mathbf{x}}^T \mathbf{y}^{(i)} - b^{(i)}\right)^2$.

$$\nabla f(x) = \sum_{i=1}^{n} \nabla \left(\underbrace{x^T y^{(i)} - b^{(i)}}\right)^2 \quad \text{(linearity)}$$

$$= 2\left(x^T y^{(i)} - b^{(i)}\right) \cdot \underbrace{\nabla\left(x^T y^{(i)} - b^{(i)}\right)}_{y^{(i)}}$$

$$\sum_{i=1}^{n} \underbrace{2\left(x^T y^{(i)} - b^{(i)}\right)}_{\text{scalar}} \cdot \underbrace{y^{(i)}}_{\text{vector}} = [\;]$$

Matrix view:

$$x^T Y^T Y x \qquad (x+\Delta)^T Y^T Y (x+\Delta)$$

$$\frac{x^T Y^T Y x + 2x^T Y^T Y \cancel{\Delta} + \Delta^T Y^T Y \Delta}{\Delta}$$

$$\left\| \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}}_{Y} \begin{bmatrix} x \end{bmatrix} - \underbrace{\begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(n)} \end{bmatrix}}_{B} \right\|_2^2$$

$$f(x) = \|Yx - b\|_2^2$$

$$\nabla (x^T M x) = 2Mx$$

quadratic form

$$f(x) = (Yx-b)^T(Yx-b) = \underbrace{x^T Y^T Y x}_{2Y^TYx} \underbrace{- 2x^T Y^T b}_{-2Y^Tb} + \underbrace{b^T b}_{0}$$

$$\nabla f(x) = 2Y^T Y x - 2Y^T b = 2Y^T(Yx - b)$$

$x^T M x = \sum_{i,j=1}^{d} x_i x_j M_{i,j}$ $\qquad$ $\frac{\partial f}{\partial x_1}\left(\sum_{i,j} x_i x_j M_{i,j}\right)$

Greedy approach: Given a starting point $\mathbf{x}$, make a small adjustment that decreases $f(\mathbf{x})$. In particular, $\mathbf{x} \leftarrow \mathbf{x} + \eta\mathbf{v}$ and $f(\mathbf{x}) \leftarrow f(\mathbf{x} + \eta\mathbf{v})$.

$v = -\nabla f(x)$

$\frac{d}{dx} x^2 = 2x$

$\frac{(x+\Delta)^2 - x^2}{\Delta}$ $\qquad$ $\frac{-\cancel{x^2} + (\cancel{x^2} + 2\Delta x + \Delta^2)}{\Delta} = 2x + \Delta$

What property might I want in $\mathbf{v}$?

$\nabla f(x)^T v \longrightarrow$ negative

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t} = \nabla f(\mathbf{x})^T \mathbf{v}.$$

$\underbrace{f(x+tv) - f(x)}_{\text{negative}} \approx t\,\nabla f(x)^T v$

$t\,\nabla f(x)^T(-\nabla f(x))$

$= -t\,\nabla f(x)^T \nabla f(x) = -t\,\|\nabla f(x)\|_2^2$

GRADIENT DESCENT

$v = -\nabla f(x^{(i)})$

Prototype algorithm:

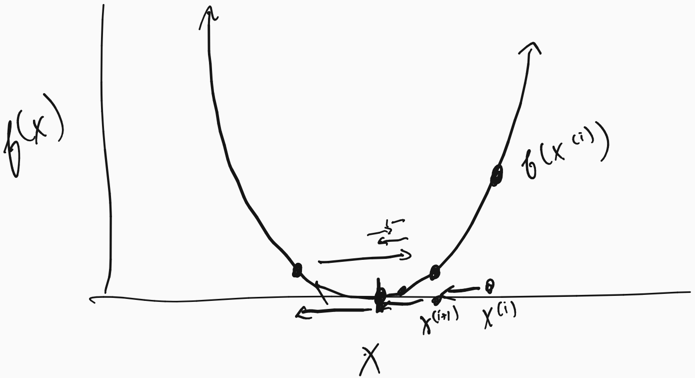
- Choose arbitrary starting point $\underline{\mathbf{x}}^{(1)}$.
- For $i = 1, \ldots, T$: step size
  - $\underline{\mathbf{x}^{(i+1)}} = \underline{\mathbf{x}^{(i)}} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\mathbf{x}^{(t)}$.

η is a step-size parameter. Needs to be chosen ahead of time or adapted on the go.

$- [100000, 0000]$

$\underline{\min \ \nabla f(x)^T v}$

Example in one dimension:



Why is gradient descent sometimes called "steepest descent"?

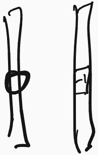
Claim (Gradient descent = Steepest descent)

$$\frac{-\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2} = \arg\min_{\mathbf{v}, \|\mathbf{v}\|_2 \leq 1} \nabla f(\mathbf{x})^T \mathbf{v}$$

Note: We could have chosen to restrict $\mathbf{v}$ using a different norm. What if we had restricted $\|\mathbf{v}\|_1 \leq 1$? $\|\mathbf{v}\|_\infty \leq 1$? These choices lead to variants of generalized steepest descent..

$\|v\|_1 \leq 1$

$\min_{v : \|v\|_2 \leq 1} \nabla f(x)^T v$



Forward Stagewise Regression

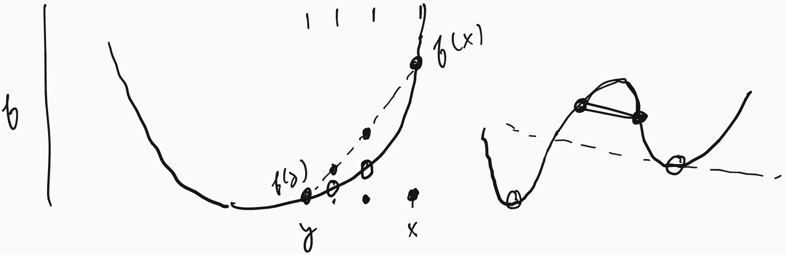
In general, gradient descent can be proven to converge (and we understand how quickly it converges) for convex functions.

Definition (Convex)

A function f is convex iff for any $\mathbf{x}, \mathbf{y}, \underbrace{\lambda \in [0,1]}$:

$$(1-\lambda)\cdot f(\mathbf{x}) + \lambda \cdot f(\mathbf{y}) \geq f\left(\underbrace{(1-\lambda)\cdot \mathbf{x} + \lambda \cdot \mathbf{y}}\right)$$

straight line



Definition (Convex)

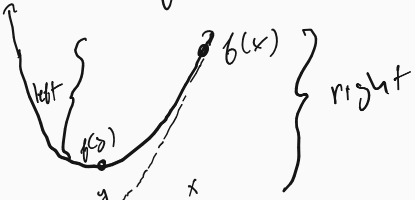
A function f is convex iff for any $\mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}+\mathbf{z}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{z}$$

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \nabla f(\mathbf{x})^T(\mathbf{x}-\mathbf{y})$$

$z = y - x$ $\quad f(y) \geq f(x) + \nabla f(x)^T(y-x)$

$$f(y) - f(x) \geq \nabla f(x)^T(y-x)$$



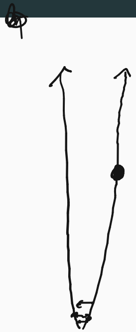
Definition (Convex)

A function f is convex iff for any $\mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \nabla f(\mathbf{x})^T(\mathbf{x} - \mathbf{y})$$

Assume:

- f is convex. function
- Lipschitz ~~gradients~~: for all $\mathbf{x}$, $\|\nabla f(\mathbf{x})\|_2 \leq G$.
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

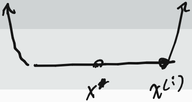


Gradient descent:

- Choose number of steps T.
- $\eta = \frac{R}{G\sqrt{T}}$
- For $i = 1, \ldots, T$:
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}^{(i)}} f(\mathbf{x}^{(i)})$.
- Alternatively, return $\hat{\mathbf{x}} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{x}^{(i)}$.

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$, *then* $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.



$$\underbrace{\|x^{(i+1)} - x^*\|_2^2} = \|x^{(i)} - \eta \nabla f(x^{(i)}) - x^*\|_2^2$$

$$= \|x^{(i)} - x^*\|_2^2 + \eta^2 \|\nabla f(x^{(i)})\|_2^2 - \underbrace{2\eta \nabla f(x^{(i)})(x^{(i)} - x^*)}$$

$$\underbrace{\nabla f(x^{(i)})(x^{(i)} - x^*)} = \frac{\|x^{(i)} - x^*\|_2^2 - \|x^{(i+1)} - x^*\|_2^2}{2\eta} + \frac{\eta}{2} \underbrace{\|\nabla f(x^{(i)})\|_2^2}_{\leq G^2}$$

$$f(x^{(i)}) - f(x^*) \leq \nabla f(x^{(i)})(x^{(i)} - x^*)$$

$$f(x^{(i)}) - f(x^*) \leq \frac{\|x^{(i)} - x^*\|_2^2 - \|x^{(i+1)} - x^*\|_2^2}{2\eta} + \frac{\eta}{2} G^2 \quad \text{for all } i$$

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

$$\eta = \frac{R}{G\sqrt{T}}$$

$$f(x^{(i)}) - f(x^*) \leq \frac{\|x^{(i)} - x^*\|_2^2 - \|x^{(i+1)} - x^*\|_2^2}{2\eta} + \frac{\eta}{2}G^2 \quad \text{for all } i$$

$$\sum_{i=1}^{T} f(x^{(i)}) - f(x^*) \leq \sum_{i=1}^{T-1} \frac{\|x^{(i)} - x^*\|_2^2 - \|x^{(i+1)} - x^*\|_2^2}{2\eta} + \frac{T\eta}{2}G^2$$

$$= \frac{\|x^{(1)} - x^*\|_2^2 - \|x^{(T)} - x^*\|_2^2}{2\eta} - \frac{T\eta}{2}G^2$$

$$\leq \frac{R^2}{2\eta} + \frac{T\eta}{2}G^2 \leq \frac{RG}{2\sqrt{T}} + \frac{RG}{2\sqrt{T}}$$

$$\sum_{i=1}^{T} f(x^{(i)}) - f(x^*) \leq RG\sqrt{T}$$

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$. $\frac{1}{\varepsilon}$ $\log(1/\varepsilon)$

$$\underbrace{\sum_{i=1}^{T} f(x^{(i)}) - f(x^*)} \leq RG\sqrt{T}$$

$$\frac{1}{T}\left[\sum_{i=1}^{T} f(x^{(i)}) - f(x^*)\right] \leq \frac{RG}{\sqrt{T}} \leq \varepsilon$$

when $T \geq \frac{R^2G^3}{\varepsilon^2}$

$$f\Big(\underbrace{\frac{1}{T}\sum_{i=1}^{T} x^{(i)}}_{\hat{x}}\Big) - f(x^*) \leq \underbrace{\frac{1}{T}\left[\sum_{i=1}^{T} f(x^{(i)}) - f(x^*)\right]}$$

or $\to \hat{x}$

$$\boxed{f(\hat{x}) - f(x^*) \leq \varepsilon}$$

$$\min_i f(x^{(i)}) - f(x^*) \leq \frac{1}{T}\sum_{i=1}^{T} f(x^{(i)}) - f(x^*)$$

Instead of a single function f to minimize, assume we have an unknown and changing set of objective functions:

$$f_1, \ldots, f_T.$$

- At each time step, choose $\mathbf{x}^{(i)}$.
- f_i is revealed and we pay cost $f_i(\mathbf{x}^{(i)})$
- **Goal**: Minimize $\underbrace{\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})}$.

$\|XW - B\|$

$x^{(1)} \ldots x^{(T)}$

$\|Ax-b\|_2^2$

Objective: Choose $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$ so that:

"regret"

$$\underbrace{\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})} \leq \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right] + \Delta.$$

We want to compete with the best fixed solution in hindsight.

opt = x^*

Assume:

- Lipschitz ~~gradients~~ function: for all $\mathbf{x}$, i, $\|\nabla f_i(\mathbf{x})\|_2 \leq G$.
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

Online Gradient descent:

- Choose number of steps T.
- $\eta = \frac{R}{G\sqrt{T}}$
- For $i = 1, \ldots, T$:
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)})$
- Play $\mathbf{x}^{(i+1)}$.

Claim (OGD Regret Bound)

After T steps, $\Delta = \left[\underline{\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})}\right] - \left[\underline{\sum_{i=1}^{T} f_i(\mathbf{x}^*)}\right] \leq RG\sqrt{T}$

$$\underbrace{\nabla f_i(x^{(i)})(x^{(i)} - x^*)} \leq \frac{\|x^{(i)} - x^*\|_2^2 - \|x^{(i+1)} - x^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$f_i(x^{(i)}) - f_i(x^*) \leq$$

$$\underbrace{\sum_{i=1}^{T} f_i(x^{(i)}) - f_i(x^*)} \leq \frac{\|x^{(1)} - x^*\|_2^2 - \|x^{(T)} - x^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$\leq \frac{R^2}{2\eta} + \frac{\eta G^2}{2} = \underline{\underline{RG\sqrt{T}}}$$

$$\frac{1}{T}\sum_{i=1}^{T} f_i(x^{(i)}) - f_i(x^*) \leq \frac{RG}{\sqrt{T}}$$

Claim (OGD Regret Bound)

After T steps, $\Delta = \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] - \left[\sum_{i=1}^{T} f_i(\mathbf{x}^*)\right] \leq RG\sqrt{T}$

Recall the machine learning setup. In empirical risk minimization, we can typically write:

$$f(\mathbf{x}) = \sum_{j=1}^{n} f_j(\mathbf{x})$$

where f_i is the loss function for a particular data point.

Linear regression:

$$f(\mathbf{x}) = \sum_{j=1}^{n} (\mathbf{x}^T \mathbf{y}^{(j)} - b^{(j)})^2$$

Pick random $j \in 1, \ldots, n$:

$$\mathbb{E}\left[\nabla f_j(\mathbf{x})\right] = \nabla f(\mathbf{x}).$$

But $\nabla f_j(\mathbf{x})$ can be computed in a $1/n$ fraction of the time!

Main idea: Use random approximate gradient in place of actual gradient.

Trade slower convergence for cheaper iterations.

Assume:

- Lipschitz ~~gradients~~ functions: for all $\mathbf{x}$, j, $\|\nabla f_j(\mathbf{x})\|_2 \leq \frac{G'}{n}$.
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

Stochastic Gradient descent:

- Choose number of steps T.
- $\eta = \frac{D}{G'\sqrt{T}}$
- For $i = 1, \ldots, T$:
    - Pick random $j_i \in 1, \ldots, n$.
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_{j_i}(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{x}^{(i)}$

Claim (SGD Convergence)

After $T = \frac{R^2 G'^2}{\epsilon^2}$ iteration:

$$\mathbb{E}\left[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)\right] \leq \epsilon.$$

Claim (SGD Convergence)

After $T = \frac{R^2 G'^2}{\epsilon^2}$ iteration:

$$\mathbb{E}\left[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)\right] \leq \epsilon.$$

Claim (SGD Convergence)

After $T = \frac{R^2 G'^2}{\epsilon^2}$ iteration:

$$\mathbb{E}\left[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)\right] \leq \epsilon.$$

Number of iterations for error ϵ:

- **Gradient Descent**: $T = \frac{R^2G^2}{\epsilon^2}$.
- **Stochastic Gradient Descent**: $T = \frac{R^2G'^2}{\epsilon^2}$.

Always have $G \leq G'$:

$$\|\nabla f(x)\|_2 \leq \|\nabla f_1(x)\|_2 + \ldots + \|\nabla f_n(x)\|_2 \leq n \cdot \frac{G'}{n} = G'.$$

Fair comparison:

$$\frac{R^2G'^2}{\epsilon^2} = n \cdot \frac{R^2G^2}{\epsilon^2}$$